Song Genre Predictor based on Spotify Data

Data Mining I Project Report

presented by
Team 1
Miguel Samaniego
Matriculation number 1980439
Elizaveta Nosova
Matriculation number 1983805
Nico Sharei
Matriculation number 1986818
Julian Ament
Matriculation number 1981511
Artem Bisliouk
Matriculation number 1978986
Jannik Kranz
Matriculation number 1981766

submitted to the
Data and Web Science Group
Prof. Dr. Heiko Paulheim
University of Mannheim

08.12.2023

Contents

1 Business Understanding

Spotify is one of the leading audio streaming services with over 500 million users worldwide. Among other features, it excels by its system of personalized recommendations for artists, tracks, and playlists, largely based on genres. Appropriate recommendations that match the user's preferred genres improve the overall customer experience and increase user loyalty to the platform. Hence, the task of categorizing tracks by genres is important for gaining a significant competitive advantage.

Traditionally, genre assignment depended on record labels, music critics, and sometimes artists' own identification, which involved a certain degree of subjectivity. Since it is a classification task, nowadays a more modern approach using machine learning algorithms is implemented to make the process more efficient and the results more accurate. Genres are treated as labels (a target variable) while each track is assessed on a set of measurable attributes.

The goal of this project is to create a classifier and see how accurately it can predict song genres. Taking a dataset from Spotify [Pandya, 2022], which is already using machine learning algorithms for these purposes, can help assess if the resulting model can be considered apt for a large-scale business or is more appropriate for a smaller audio streaming market player.

2 Data Understanding

The dataset consists of 114.000 songs and encompasses 114 genres. The data regarding each genre is already balanced, containing 1.000 samples per genre. However, during the exploratory data analysis, around 16.000 entries of the same songs appearing with different genres were found.

With respect to missing values, the quality of the data is high, as there is only one row with missing entries. On the other hand, the quality regarding the target variable is questionable. One good example of this is classical music composer Johann Sebastian Bach. There are several instances where his musical pieces appeared labelled as *german* while on other instances his work is labelled as *classical*.

Regarding the attributes, an overview of all the features and their data types are presented in Table 1. To get a better understanding on the relationships between the features, a correlation analysis was implemented (see Figure 1). The figure shows that the features *loudness* and *energy* as well as *acousticness* and *energy* have a high correlation (0,75 and -0,71 respectively), while *acousticness* and *loudness* have a moderate correlation (-0,53). This shows the close relation between these three features. The other features show a relatively low correlation.

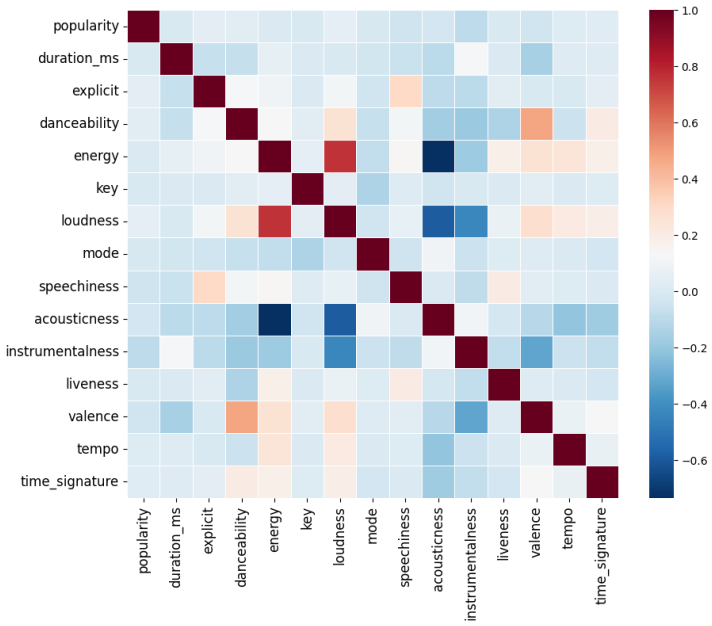


Figure 1: Correlation analysis.

The attributes represent different features of a song, ranging from pure objective features, like duration, to more complex features, like valence. The following description will help to gain a better understanding of the data. The *key* feature represents the standard pitch of a song, while *time signature* specifies how many bars are in each measure. *Speechiness* measures the amount of spoken words in a song, the closer to 1 the song is, the higher the amount of spoken words. Other features like *instrumentalness* is measured by the amount of non vocal parts of a song, 1 indicating that there are no vocal parts and 0 indicating only vocal parts, like in a rap song. Other features include background noise, like *liveliness*, which measures if there is a audible life audience in the background. The feature *valence* is used as an indicator on how positive or happy a song is, 1 indicating very positive songs and 0 indicating the least positive songs.

Attribute	Data Type	Range
track_id	Integer	[1;89741]
artists	String	*Textual*
album_name	String	*Textual*
track_name	String	*Textual*
popularity	Integer	[0;100]
duration_ms	Integer	[8586;5237295]
explicit	Boolean	[True, False]
danceability	Double	[0;1]
energy	Double	[0;1]
key	Integer	[0-11]
loudness	Double	[-49.5;4.532]
mode	Integer	[0,1]
speechiness	Double	[0;0.965]
acousticness	Double	[0;0.996]
instrumentalness	Double	[0;1]
liveness	Double	[0;1]
valence	Double	[0;0.995]
tempo	Double	[0;243]
time_signature	Integer	[3-7]
track_genre	String	*Categorical*

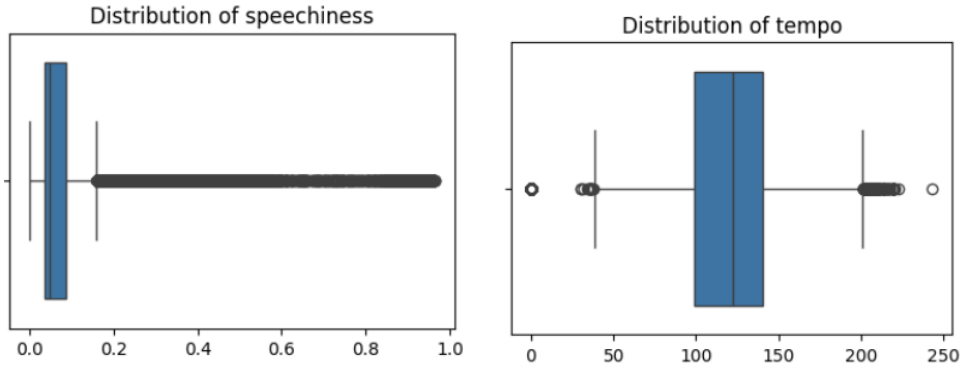
Table 1: Attributes found in the dataset.

3 Preprocessing

To work with the dataset, several preprocesssing steps, including eliminating missing values, dealing with outliers, and normalisation were carried out.

3.1 Outliers & Missing Values

Missing values were only found in one case, where the *track_name*, the *artist* and the *album_name* were missing, this song was excluded from further analysis. Outliers turned out to be an important indicator for some music genres, like the comedy genre, which includes all outliers of the feature *speechiness*, seen in in Figure 2a. Therefore after filtering for outliers the prediction of our models worsened and we decided to keep them in the training data.



(a) Distribution of speechiness. (b) Distribution of tempo.

Figure 2: Comparison between the attributes *speechiness* and *tempo*. It is observable that while the outliers of *tempo* are equally distributed around the mean, the outliers of *speechiness* are fully located towards the maximum.

The dataset doesn't contain any invalid values or inconsistencies among the attribute values. Nevertheless, different units of measurement of several features needed to be addressed to enable distance-based clustering and classification methods, such as k-nearest neighbors, support vector machine and hierarchical agglomerative clustering. The standard scaler tool was used for normalization.

3.2 Reducing Genres

A reduction in the number of values for our target variable was implemented due to the complication in interpreting the results when working with the original 114 genres. Additionally, there was great similarity between several genres in the original dataset. Such examples include: *techno*, *minimal-techno*, *detroit-techno*, etc.

To perform this reduction of genres, a hierarchical clustering algorithm was utilized. Before running the hierarchical clustering, the dataset was grouped by genre and the mean of the numerical values was computed. Additionally, the values were scaled using a standard scaler. After doing so, this data was fed to the algorithm and a dendrogram was obtained as a result. This method enabled us to understand the relationship between different genres and allowed us to group the 114 genres into only 18.

3.3 Duplicates

Songs that were entirely duplicate were checked and eliminated from the Spotify song dataset, preventing potential skewing of the analysis or model training. Han-

dling multiple entries of songs across different genres involved two implemented solutions. One approach involved random selection, whereas the other method employed a similarity-based selection process.

For the similarity-based selection, we first grouped songs by genre and created a vector, consisting of the mean for each attribute. Subsequently, we computed the distance between the feature vector of each song and the respective genre mean vector. By selecting the genre with the smallest distance to the song, we effectively categorized it based on its similarity to genre attributes. This method aimed to allocate songs more accurately across genres, addressing the challenge posed by multiple entries for a single song across different genres in the Spotify dataset.

3.4 Feature Generation

As the features *track_name*, *artists* and *album_name* do contain textual data, we are enabled to apply Text Mining techniques to extract additional value from them. The Term Frequency-Inverse Document Frequency (TF-IDF) was applied on each attribute. The best results appeared for the *artist* attribute.

Feature	Score
los	0.003832
mc	0.002628
pritam	0.002394
wolfgang	0.001895
zimmer	0.001449

Table 2: Top 5 most effective features generated by the TF-IDF methodology.

Table 2 shows that we can easily interpret which information is encoded by the newly generated features. For instance, *los* is a spanish term and a high value can serve as an indicator for music belonging to the category *latin*. On the other hand, *Zimmer* could have originated from music belonging to the artist *Hans Zimmer*, who primarily composes instrumental music. A downside which shouldn't be ignored is that the majority of artists possesing the name *Zimmer* produce german music. In this case, german songs may be misclassified as instrumental and vice versa.

4 Data Mining

Our data mining approach consists of several different steps (see Figure 3). For these steps we tried out a 70/30 and a 80/20 split between the training and test data.

Additionally, hyperparameter optimization was implemented by implementing 5-fold cross validation on the training set and optimizing for the weighted F1-score. It is important to note that stratified sampling was used to ensure that the minority classes will still get represented in the training set after doing the data splits.

Figure 3: General procedure for each machine learning approach.

There were a total of five different machine learning approaches implemented:

1. k Nearest Neighbors
2. Naive Bayes
3. Decision Tree
4. Random Forest
5. Support Vector Machine

4.1 Baseline

To be able to compare the effectiveness of the machine learning methods, two different baselines were used. The first one was the simplest one, as it only involved predicting the majority class.

Due to the simplicity of the majority baseline, an alternative, more intricate approach was implemented by using a set of rules to predict the genre of a song. These rules were obtained by looking at the outlier values of different features and finding out which genres had most of these occurrences. For example, if a song had a *speechiness* value above 0.75, it would be classified as *comedy*, since this was the only genre with such samples.

4.2 Hyperparameter Tuning

The tuning of the hyperparemeters of the models was carried out using sklearn's *GridSearchCV* function. The value ranges for the hyperparameters always included the default values. This was done to ensure that the tuning yielded a better performance than the default version of the model. An example of the parameter

tuning performed for the Random Forest classifier on the dataset with 18 genres is presented in Table 3.

Parameter	Range	Best value
n_estimators	[100, 200, 300]	300
max_features	[None, sqrt, log2]	sqrt
min_samples_leaf	[1, 10, 50, 100]	1
min_samples_split	[2, 10, 50, 100]	2
max_depth	[None, 50, 100]	None

Table 3: Hyperparameter ranges used for Random Forest classifier.

It is important to note that for most of these parameters, the optimal value often resulted in being the default one. The only exception was with the *n_estimators* parameters, where a bigger value than the default of 100 was found to be the optimal one for the Random Forest model.

4.3 Balancing

Removing duplicates from the original data set introduced considerable unbalance. While the majority classes still had approximately 1000 instances, the count of minority classes dropped to 245. The clustered data set with 18 classes displays an even larger unbalance with the ratio of majority to minority class instances of 10894 to 990. Since unbalance is highly likely to have a negative impact on models' performance, it was decided to conduct an experiment with balancing.

RandomOverSampler and *RandomUnderSampler* from the *imblearn* library were applied to both 114-class and 18-class versions of the train data set after the train/test split. In none of the cases training a model on a balanced data set outperformed the initial results, and in some cases, particularly undersampling for kNN and both balancing approaches for Random Forest, considerably hindered F1 score by 0.06-0.1.

4.4 Evaluation

The evaluation against the baseline showcased significant improvements (see Table 4). In the simplified dataset, performance surged from 0.09 to 0.66, a 7-fold increase, when comparing the base line model with the random forest model. The full-class dataset displayed an even more substantial rise, soaring from 0.03 in the baseline to 0.61, a 20-fold enhancement achieved by the Random Forest model.

Model	114 genres (F1)	18 genres (F1)	Improvement (%)
Baseline	0.03	0.09	200
Naïve Bayes	0.18	0.25	38.89
SVM	0.33	0.45	36.36
k NN	0.35	0.5	42.86
Decision Tree	0.37	0.47	27.03
Random Forest	0.61	0.66	8.2

Table 4: Final F1-score for the different machine learning methods.

The Random Forest classifier emerged as the top-performing algorithm, consistently achieving a high F1-Score of 0.66. Its performance showcases its suitability for accurately assigning genres to songs, making it the optimal choice among the classifiers tested.

Figure 4 displays the resulting confusion matrix for the Random Forest model. It is observable that highly energetic and loud genres such as *techno*, *edm* and *rock* are proportionally often misclassified for each other while *folk* and *latin* may have often been misclassified due to similarity in acousticness and valence.

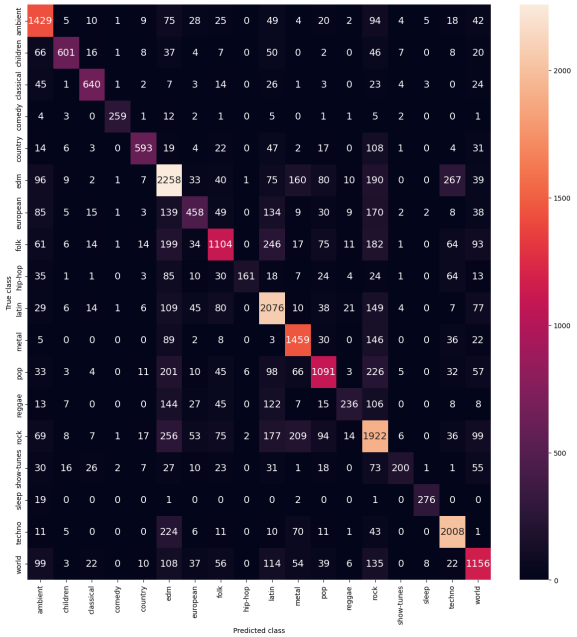


Figure 4: Confusion Matrix of the Random Forest model.

Regarding the importance of the features, as seen in Figure 5, all these values are very relevant for the decision trees in the random forest, therefore highly correlating genres do share similar paths and can easily be misclassified if a rather deep, less expressive condition fits slightly more to the wrong class. Especially when considering the highly flexible nature of music.

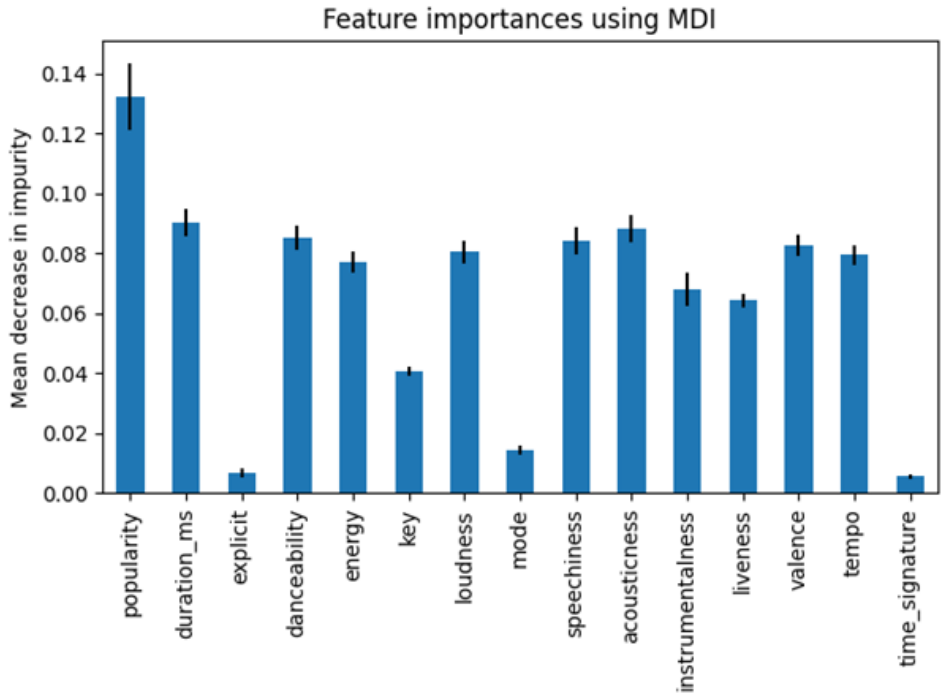


Figure 5: Feature importance using MDI.

The performance of the Gaussian Naive Bayes resulted in the worst F1-Score. The satisfaction of independence assumption, which serves as the base of this method, means we remove the highly correlated features from consideration.

An exploration of the feature distributions revealed that most of the final scaled features, crucial for our prediction, do not adhere to a normal distribution seen in Figure 6. We evaluated the feature distributions using the Kolmogorov-Smirnov (KS) test and P-value. The average KS test statistic is 0.12, and the average P-value is 3.9×10^{-86}. These results suggest a very weak correspondence between the distributions of our features and a normal distribution which is a negative factor for the Gaussian Classifier. This resulted in our relatively small F1-Score.

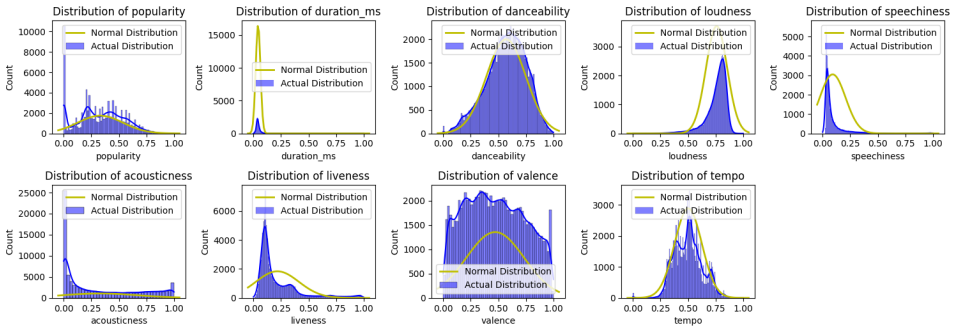


Figure 6: Feature Distributions.

5 Results

Performance of all models elaborated in this project is satisfactory against both majority and rule-based baselines, Random Forest being the best performing one. Still, business implementation of this model is questionable. In the context of our business task, obtained result would mean that every third track would be mislabeled. As a consequence, user satisfaction with the recommendation system can be average, and artists that put their tracks on the platform might not get the full potential of outreach to appropriate audience.

Hence, current results can be used as a base for further improvement. First, addressing mislabeled records in the training set with the help of music industry experts can be attempted. Then, instead of removing duplicates with different genres, the task can be rescaled to a multi-label classification problem. Additionally, extracting more diverse feature data, such as lyrics through text mining, could offer deeper insights and potentially improve our model's predictive capabilities.

References

[Pandya, 2022] Pandya, M. (2022). Spotify tracks dataset.